

Alexander D. Pencu (*pro hac vice*)  
adp@msf-law.com

Christopher J. Major (*pro hac vice*)  
cjm@msf-law.com

Jeffrey P. Weingart (*pro hac vice*)  
jpw@msf-law.com

**MEISTER SEELIG & FEIN PLLC**

125 Park Avenue, 7<sup>th</sup> Floor  
New York, NY 10017  
Telephone: (212) 655-3500  
Fax: (646) 539-3649

Patricia L. Peden (CA Bar No. 206440)  
patricia@warrenkashwarren.com

**WARREN KASH WARREN LLP**

2261 Market Street No. 606  
San Francisco, CA 94114  
Telephone: (415) 895-2940  
Fax: (415) 895-2964

*Attorneys for Plaintiff*

**UNITED STATES DISTRICT COURT  
NORTHERN DISTRICT OF CALIFORNIA  
SAN FRANCISCO DIVISION**

LYNWOOD INVESTMENTS CY LIMITED,

Plaintiff,

vs.

MAXIM KONOVALOV, IGOR SYSOEV,  
ANDREY ALEXEEV, MAXIM DOUNIN, GLEB  
SMIRNOFF, ANGUS ROBERTSON, NGINX,  
INC. (BVI), NGINX SOFTWARE, INC., NGINX,  
INC. (DE), BV NGINX, LLC, RUNA CAPITAL,  
Inc., EVENTURE CAPITAL PARTNERS II, LLC  
and F5 NETWORKS, INC.,

Defendants.

Case No. 3:20-CV-03778-MMC

**DECLARATION OF NICHOLAS  
ELLISON IN SUPPORT OF  
PLAINTIFF'S MOTION TO COMPEL  
NETFLIX, INC. TO COMPLY WITH  
PLAINTIFF'S JUNE 10, 2025  
SUBPOEANA**

Date:

Time: 9:00 a.m.

Courtroom 7, 19th Floor

The Honorable Maxine M. Chesney

1 I, Nicholas Ellison, hereby declare as follows:

2 1. I am over 21 years of age. This declaration is based upon my personal knowledge. I  
3 am familiar with the contents of this declaration and could testify under oath to the facts set forth  
4 herein if called as a witness.

5 2. I am a Managing Director in the Cyber and Data Resilience practice at Kroll, LLC  
6 (“Kroll”), a financial and risk advisory firm based in New York City. I am based in Kroll’s London  
7 office and have been with Kroll since August 2022.

8 3. I have prepared this declaration in response to an instruction from Meister Seelig &  
9 Fein PLLC, counsel for Plaintiff Lynwood Investments CY Limited (“Lynwood”). I have been asked  
10 to provide this declaration to assist the United States District Court for the Northern District of  
11 California (“Court”) in determining questions related to the discovery Lynwood seeks from Netflix,  
12 Inc. (“Netflix”) pursuant to Lynwood’s June 10, 2025 subpoena to Netflix (“Subpoena”).  
13 Specifically, I have been asked to provide background information and analysis related to version  
14 control systems, a software development tool that manages and tracks changes to source code over  
15 time.

16 4. I respectfully submit this declaration in connection with Lynwood’s motion to compel  
17 Netflix to comply with the Subpoena.

18 5. My qualifications to opine on the software development process and version control  
19 systems are contained in my curriculum vitae, which is attached hereto as Appendix A.

20 6. For background purposes, I have reviewed the Second Amended Complaint filed by  
21 Lynwood on April 7, 2025 (“SAC” or “Complaint”), the Subpoena, the transcript of the March 7,  
22 2025 Case Management Conference held in this action (“CMC”), and Netflix’s June 24, 2025 and  
23 July 28, 2025 responses to the Subpoena.

24 7. I express no view on the merits of this action, but accept Plaintiff’s allegations for  
25 purposes of this Declaration. My understanding is that this copyright infringement action concerns  
26 the development of a software product known as NGINX Plus, Plaintiff’s ownership of the copyright  
27 on NGINX Plus, and Defendants’ infringement. *See* SAC, ¶¶14, 477. I also understand Plaintiff to  
28 allege that Defendants Sysoev, Konovalov, and Smirnoff are former employees of a Russian

1 technology company, Rambler Internet Holding LLC (“Rambler”), and that while employed by  
 2 Rambler, these defendants used Rambler’s resources and infrastructure to develop proprietary source  
 3 code and modules for what became NGINX Plus. *Id.*, ¶¶6-7. I further understand that Plaintiff  
 4 alleges that the commercial NGINX code that was developed by Sysoev, Konovalov and Smirnov  
 5 while they were employed by Rambler and subsequently incorporated into NGINX Plus is referred  
 6 to as “Pre-Exit NGINX Plus.” *Id.*, ¶2.

### 7 **Software Development and Version Control Systems**

8 8. In software development, “version control” is a system that maintains different  
 9 versions of a set of files in a centralized database, known as a “repository,” that stores the complete  
 10 collection of files and folders for a codebase, along with revision history. Version control systems  
 11 allow developers to track and manage changes to a file or a set of files over time, and are particularly  
 12 useful when there are multiple developers working on the same project. Version control systems  
 13 facilitate collaboration among a group or team of developers by, for example, allowing developers  
 14 to write or modify code for the same project simultaneously without overwriting each other’s work.

15 9. Essentially, a version control system records snapshots of a project at different points  
 16 in time. In theory, the information available through a version control system allows developers to  
 17 track changes to files over time, view modifications, and identify when and by whom changes were  
 18 made.

19 10. Changes made through a version control system are managed using the following key  
 20 concepts:

- 21 a. Repository: A repository is the centralized database, normally a set of hidden files in  
 22 the folder that stores the complete collection of files and folders for a codebase.
- 23 b. Commits: A commit is a snapshot of a software development project at a specific  
 24 point in time. Each commit has a message allowing the developer to describe the  
 25 changes, including when a file or directory was first committed to the repository (the  
 26 “commit date”). As discussed in more detail below, while commits can provide useful  
 27 historical records, they are not tamper-proof, and developers can manipulate commit  
 28

1 information, including by modifying the commit date and author information. Also,  
2 developers can move code between repositories with or without preserving history.

3 c. Branches: A branch is a separate line of development. Developers can create branches  
4 to work on features or debug independently without changing the main codebase.  
5 Once the work is complete, the developer can merge the “branch” back into the main  
6 codebase.

7 d. Pull Requests or Merge Requests: In team settings, developers may submit a “pull  
8 request,” also referred to as “merge requests,” to propose their changes before the  
9 changes are merged back to the main codebase. This allows developers to work  
10 collaboratively and review work for quality control.

11 11. There are two main types of version control systems, a Centralized Version Control  
12 System and a Distributed Version Control System. In a Centralized Version Control System, all  
13 changes and modifications are stored on a central server, or a shared repository, and developers must  
14 connect to the central server to make changes to files, *i.e.* “commits,” or to retrieve updates.  
15 Examples of Centralized Version Control Systems include Subversion (SVN) and Perforce. The  
16 “centralized” nature of these systems presents potential risks, the primary being the single point of  
17 failure that the centralized server represents. For example, because all users work with the same  
18 repository and commit to the same branch, if the centralized server fails, no changes can be made or  
19 saved to the project until the server comes back online. Moreover, if the central server becomes  
20 corrupted, and proper backups have not been maintained, there is the risk of losing the entire history  
21 of the project stored on the central server.

22 12. In a Distributed Version Control System, each developer has a full copy of the entire  
23 version history on their local machine. This allows developers to work offline from any location,  
24 which allows for more flexible collaboration and avoids the risk of loss present in Centralized  
25 Version Control Systems. Unlike a Centralized Version Control System, the decentralized nature of  
26 a Distributed Version Control System allows users to commit, branch, or merge changes locally  
27 without reliance on a central server and contributors can work on the same codebase without being  
28 on the same network. Examples of Distributed Version Control Systems include Git and Mercurial.

1 Most modern software teams use GitLab or GitHub, which is a web-based platform that uses the Git  
2 technology.

### 3 **Commit Information Manipulation**

4 13. As mentioned above, although the commit information stored on a version control  
5 system is intended to provide users with information to track the historical development of a file or  
6 codebase, version control systems are not intended to act as an anti-tamper mechanism.

7 14. Below, I include a few examples of how commit information can be tampered with  
8 to illustrate scenarios where commit information stored on a version control system can be  
9 misrepresented, manipulated, or used to obscure the development history of a file, including  
10 timestamps and authorship.

- 11 a. **Delayed Commitment of Code:** Version control systems do not enforce commit  
12 frequency, and developers can write and develop source code without formally  
13 committing it to a repository. In this case, the code would simply remain on the  
14 developer's local device until such time the developer decided to commit the code to  
15 the repository, which could be months or years after the code was written. I  
16 understand Plaintiff alleges that beginning in or around 2009, Defendants started to  
17 develop, but not commit, the Pre-Exit NGINX Plus code developed at Rambler. SAC,  
18 ¶¶127-128, 311. It is also my understanding that Defendants stored Pre-Exit NGINX  
19 code developed with Rambler's resources and infrastructure on personal servers that  
20 were removed from Rambler after Defendants Sysoev, Konovalov, and Smirnov left  
21 Rambler. *Id.*, ¶311. Under this scenario, the Pre-Exit NGINX Plus code developed  
22 at Rambler could have remained on Defendants' local devices for an extended period  
23 of time before being committed to a repository, *e.g.*, after Sysoev, Konovalov, and  
24 Smirnov left Rambler, thus creating the false appearance of a later development date.
- 25 b. **Copying Code From a Previous Repository:** Plaintiff alleges that Defendants used  
26 Rambler's infrastructure to develop, test, and store proprietary code later  
27 commercialized as NGINX Plus. *See*, SAC, ¶475. Plaintiff also alleges that  
28 Defendants deleted, removed, or destroyed the Rambler servers used to develop Pre-

1 Exit NGINX Plus. *Id.*, ¶¶308-318. Based on these allegations, it is plausible that the  
2 repository of Pre-Exit NGINX Plus code developed at Rambler was copied to a local  
3 machine before the Rambler servers were decommissioned or destroyed. This  
4 process would create a new repository, but the repository with the copied code would  
5 not retain the prior version history. Under this scenario, the commit information  
6 stored on Defendants' current repository would not reflect the full development  
7 history of the stored files and codebase. The same is true if a repository is migrated  
8 to a different version control system. When code is migrated between different  
9 version control systems, it is likely that the existing commit information will be  
10 affected unless steps are taken in advance to preserve the existing data.

11 c. **Commit Timestamp Manipulation:** When a file is created or modified on a local  
12 machine, the operating system records timestamps such as "Date Created" or "Last  
13 Modified." However, when files are committed to a repository and later transferred  
14 to another computer the original timestamps do not transfer, and the new machine  
15 assigns new timestamps based on when the files were retrieved. Accordingly, if the  
16 NGINX Plus code stored in Defendants' present repository consists of files migrated  
17 from devices used at Rambler or other electronic device, the timestamps indicated on  
18 Defendants' repository may not accurately reflect the original commit date or the  
19 date(s) the code was written. Also, certain version control systems explicitly allow  
20 users to modify commit dates at the time of commit, or a user can achieve the same  
21 result by simply changing the clock settings on the device being used.

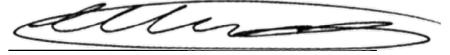
22 d. **Commit Author Manipulation:** When files are committed to a repository, they are  
23 attributed to an "Author", which is recorded as the developer who made the changes  
24 to or introduced that file. However, certain version control systems allow for the  
25 manipulation of that Author, meaning that a user can effectively change the record of  
26 who contributed what files to the repository. Changing the authorship of a commit on  
27 a repository system does not impact the file author properties on a local device. Under  
28

1 this scenario, there could be a misalignment between the repository commit record of  
2 authorship and the user edit history of files stored locally on electronic devices.

3 I declare under penalty of perjury under the laws of the United States of America that the  
4 foregoing is true and correct. This declaration was executed in London, England on August 26, 2025.

5 Dated: August 26, 2025

By:



6 Nicholas Ellison

# Appendix A





## Nick Ellison

### Managing Director

#### Contact

**E** [nick.ellison@kroll.com](mailto:nick.ellison@kroll.com)

**T** +44 (0)207 029 5498

**M** +44 (0)722 102 6619

#### Skills & Qualifications

- Holds an MEng in Systems Engineering from the University of Warwick
- Chartered Engineer (CEng)
- Chartered IT Professional (CITP)
- PRINCE2® Practitioner
- Programming languages including Python, Ruby, C++, PHP, JavaScript
- Proficient with cloud infrastructure, AWS Certified Cloud Practitioner
- Database/analytic techniques including SQL (MSSQL, PostgreSQL, MySQL), Power BI
- Data science & machine learning experience using R and Python
- Proficient with EnCase®, Axiom, Intella, Relativity, Reveal-Brainspace, and a range of eDiscovery tools
- TRM Labs certified in Digital Forensics and Cryptocurrencies, and Advanced Crypto Investigator
- Member of the British Computing Society
- Member of the Society for Computers and Law
- Member of the Institute of Engineering Technology
- Member of the Academy of Experts
- Member of the Expert Witness Institute

Nick Ellison is a Managing Director in the Cyber & Data Resilience practice, based in London. In addition, he also serves as the Lead for the Technology Expert Services team, providing expert testimony in contentious technology matters. Nick has more than 15 years of IT technology consultancy and software development experience. He has assisted clients with web and software development projects, disputed and delayed delivery of projects, fitness for purpose assessments, IT security controls, digital forensics, and cyber risk in the travel, AI, crypto, manufacturing, wholesale and distribution, financial services, charities/third sector, and hospitality sectors. He specialises in technology disputes involving software intellectual property and fitness-for-purpose.

Prior to joining Kroll, Nick founded a software development consultancy and a web development agency for the professional services sector. He has also co-founded several startups in a CTO capacity and has given expert advice on a range of technology implementation projects for third parties. Prior to that, he was a consultant at Deloitte in the Data team of the Enterprise Risk Services division.

Nick has provided training on digital evidence for prosecutors at the International Criminal Court in The Hague.

#### Representative engagements

- Instructed as an expert witness to opine on the suspected theft of Artificial Intelligence (AI) models from a Software-as-a-Service product by the previous management of a software company following an M&A transaction.
- Instructed as an expert witness, giving written testimony in court relating to whether an algorithmic trading company had properly disclosed source code compliant with a court order.
- Instructed as an expert witness to opine on the best practice of an outsourced software development team for a leading sports data aggregator. Nick's findings were presented in a CPR35 report, followed by a joint expert's report produced in coordination with the opposing side's expert.
- Instructed as an expert witness to opine on the extent to which an e-commerce platform had been successfully delivered by a supplier to the retailer, and whether bugs allegedly present in the work product would have constituted material defects.
- Instructed as an expert witness to opine on suspected source code theft in a Point-of-Sale application on an Android-powered payment terminal. Conducted black box comparison



exercise, reverse-engineering of application executables, and reviewing for evidence of potential artefacts.

- Instructed as joint expert witness to opine on suspected manipulation of email metadata relating to financial transactions.
- Supported the investigation and supplemental expert delay report for a high-profile government department software implementation dispute.
- Assisted a leading global cryptocurrency exchange with reviewing geofencing protocols, testing IT controls of both website and mobile application touchpoints, reviewing operational data logs, and making advisory recommendations for areas of improvement. Nick presented the findings to the General Counsel of the firm and leadership team.
- Provided supporting technical analysis to an expert appraisal and valuation of a suite of hospitality technology tools in relation to a high-profile family dispute which was prompting the breaking up of a business empire.
- Assessed misuse of confidential information in an airline industry payment software arbitration matter, determining how a competitor product could have been reverse-engineered.
- Led the investigation into a UK Gambling Commission compliance issue for a gambling software company, reverse engineering user activity from server logs and reviewing source code to support findings.
- Investigated suspected intellectual property theft from air ticketing booking software integrating with a travel GDS service, through review of source code and data from a limited number of intermittent backups.
- Investigated suspected source code theft for medical device firmware, by reverse engineering software binaries and developing mass processing techniques to identify potentially matching strings in included libraries.
- Led forensic investigation of email metadata across multiple inboxes from three different parties for evidence of the source of a successful payment scam targeted at a film studio.
- Assisted the forensic investigation of email and document data for an investment firm where a member of senior management was suspected of having manipulated emails for personal gain.
- Conducted forensic email examination on a range of communications with purported investors relating to a major fraud investigation of a tech unicorn.
- Led the digital forensic investigation relating to a major IPO, where key individuals had been linked with historic cybercrime



and scamming by investigative journalists. Nick led interviews with the client's staff and developed a detailed history of the major players concerned, including previously undiscovered personal and business relationships. Additional concerns were identified and shared with the client.

- Developed automated processes to identify unlicensed intellectual property in hundreds of public websites and compiled mobile applications for a leading global retailer with multiple sub-brands. Nick identified \$10m of exposure that the client was able to mitigate through a settlement deal.
- Investigated the suspected theft of pharmaceutical products from a warehouse that had been covered up through the manipulation of the in-house ERP system. Nick designed SQL analysis processes to perform additional reconciliations to identify disparities in data that had been initially covered up and presented these findings to the client.